Universidad del Valle de Guatemala
Facultad de Ingeniería
Departamento de Ciencias de la Computación
CC3067 - Redes

Laboratorio 2

Esquemas de Detección y Corrección de Errores

Stefano Alberto Aragoni Maldonado (20261)
Carol Andreé Arévalo Estrada (20461)

1. Introducción

En esta práctica se estudió e implementó diferentes mecanismos de detección y corrección de errores para poder garantizar la integridad de información al ser transmitida. Para esto, como primer paso, se seleccionaron dos algoritmos para poder detectar y corregir errores en mensajes (CRC-32 y *Código Hamming*, respectivamente). A partir de eso, se investigó y analizó el funcionamiento de los mismos para poder implementarlos en *Python* y *Java*. Finalmente, se realizó una serie de pruebas para poder determinar la robustez de estos y su capacidad para detectar y, en el caso de Código Hamming, corregir errores. Cabe destacar que esto se realizó con el objetivo de entender los retos asociados con la transmisión de mensajes en un entorno propenso a ruido y errores.

2. Objetivos

1. Comprender a detalle el funcionamiento de los algoritmos de detección y corrección.
2. Implementar los algoritmos de detección y corrección de errores.
3. Identificar las ventajas y desventajas de cada uno de los algoritmos.

3. Descripción

Como se mencionó anteriormente, en este laboratorio se implementaron los algoritmos de CRC-32 (detección de errores) y Código Hamming (detección y corrección de errores). Esto con el propósito de determinar el rendimiento de los mismos, así como ventajas y desventajas de cada uno.

El primer algoritmo implementado fue CRC-12. *Cyclic Redundancy Check (CRC)* es una técnica que permite identificar errores en mensajes transmitidos por canales muy ruidosos. Para esto, el emisor calcula y concatena un valor llamado *checksum* al final del mensaje a transmitir. Cabe destacar que el *checksum* se calcula a partir de una función polinomial definida. Por ejemplo, en el caso de CRC-32, se utiliza una función polinómica de grado 32 (P(x) = x^32 + x^26 + x^23 + x^22 + x^16 + x^12 + x^11 + x^10 + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1) (Williams, 1993).

Cada función polinomial puede ser representada de forma binaria a través de su “generador polinomial” correspondiente. Dicho número binario es posteriormente utilizado para *dividir* (división binaria en módulo 2 / XOR) el mensaje “aumentado” y así determinar el residuo (o el *checksum*). Cabe destacar que el mensaje “aumentado” es simplemente el mensaje original concatenado con *n* ceros al final, donde n representa el grado del polinomio utilizado (Patel, 2023).

Cuando el *checksum* ya fue calculado, este se concatena al final del mensaje original. A partir de esto, se puede enviar a un receptor que utiliza la misma función polinomial. Este receptor posteriormente divide el mensaje modificado (sin agregarle más bits) por el generador polinomial correspondiente. En caso el residuo de esta división sea 0, no se detectó la presencia de algún error (aunque errores pueden

pasar desapercibidos). Sin embargo, en caso contrario, esto significa que el algoritmo sí detectó algún error (Williams, 1993).

En la siguiente figura se presenta un ejemplo de Ross N. Williams (1993) del proceso de la división entre el generador polinomial y el mensaje aumentado.

Figura 1. *CRC - Cálculo de Checksum*

Fuente: Ross N. WIlliams (1993)

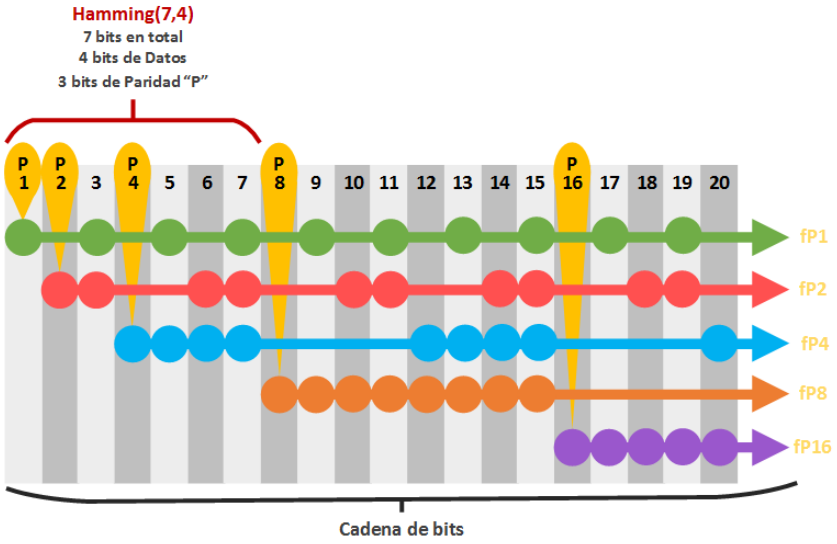
Por otro lado, el algoritmo de *Código Hamming* fue desarrollado por Richard Hamming en 1950 y es ampliamente utilizado en sistemas de comunicación y almacenamiento de datos. El Código Hamming es útil cuando la prioridad es la corrección de errores en datos pequeños, como en memorias RAM o en comunicaciones inalámbricas. Se utiliza principalmente en situaciones donde la corrección de errores es esencial y se necesita detectar y corregir errores en tiempo real. Aunque el Código Hamming puede detectar y corregir solo un bit de error en una posición determinada, su ventaja radica en su simplicidad y bajo consumo de recursos, lo que lo hace efectivo para aplicaciones con limitaciones de memoria o procesamiento.

El algoritmo de Código Hamming funciona al agregar bits de paridad a una secuencia de datos. Estos bits extra se colocan en posiciones estratégicas dentro de la secuencia para que, al llegar al receptor, se pueda detectar si ha ocurrido algún error. La ubicación de los bits de paridad se calcula de manera que su combinación indique la posición del bit erróneo y permita su corrección.

Cuando se transmite la secuencia de datos, el receptor utiliza los bits de paridad para comprobar si hay errores en la información recibida. Si se detecta un error, el algoritmo puede determinar la posición del bit incorrecto y corregirlo

automáticamente. Esto asegura que la información se recupere con precisión, brindando una mayor confiabilidad en la transmisión y almacenamiento de datos.

Figura 2. *Código de Hamming*



Fuente: (Invarato, 2017)

Para evaluar el rendimiento de ambos algoritmos implementados, se realizaron pruebas con diez tramas (mensajes) diferentes. Estas fueron procesadas por los emisores, los cuales posteriormente retornaban la trama concatenada con información adicional generada por los mismos algoritmos. Asimismo, cabe destacar que algunos resultados sufrieron modificaciones internacionales para simular errores durante la transmisión. Finalmente, estos resultados se enviaron a los receptores desarrollados en Java para comprobar si estos eran capaces de detectar y, en el caso de *Código Hamming*, corregir los errores presentes en las tramas recibidas.

4. Resultados

Después de haber implementado los correspondientes algoritmos de detección y corrección de errores, se procedió a evaluar el rendimiento de los mismos. Los resultados de estas evaluaciones se presentan a continuación.

Como primer paso, se quiso determinar si los algoritmos (CRC-32 y *Código Hamming*) no presentaban falsos positivos. Es decir, que los receptores detectaran la presencia de errores cuando no los había. Para esto, ambos emisores procesaron y modificaron las siguientes tres tramas, y posteriormente fueron enviadas a los receptores. Como se puede observar a continuación (en la *tabla 1* y *tabla 2*), ambos algoritmos lograron identificar correctamente que las tramas transmitidas no presentaban errores.

***Tabla 1**. Pruebas con CRC-32 (sin errores)*

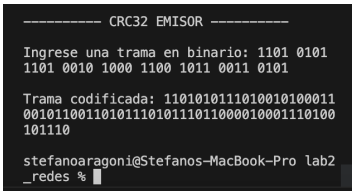
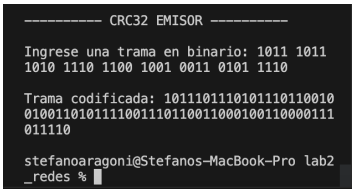
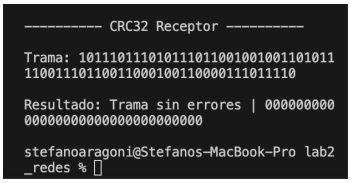
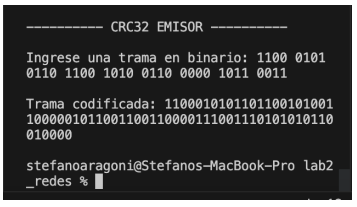
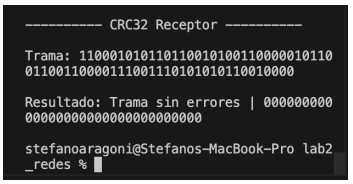
Trama (Original)	***Resultado Emisor***	***Entrada Receptor***	***Resultado Receptor***
1101 0101 1101 0010 1000 1100 1011 0011 0101		1101 0101 1101 0010 1000 1100 1011 0011 0101 1101 0111 0110 0001 0001 1101 0010 1110	 *Sin errores.*
1011 1011 1010 1110 1100 1001 0011 0101 1110		1011 1011 1010 1110 1100 1001 0011 0101 1110 0111 0110 0110 0010 0110 0001 1101 1110	 *Sin errores.*
1100 0101 0110 1100 1010 0110 0000 1011 0011		1100 0101 0110 1100 1010 0110 0000 1011 0011 0011 0000 1110 0111 0101 0101 1001 0000	 *Sin errores.*

***Tabla 2**. Pruebas con Código Hamming (sin errores)*

Trama (Original)	***Resultado Emisor***	***Entrada Receptor***	***Resultado Receptor***

1101 0101 1101 0010 1000 1100 1011 0011 0101	`========== Hamming Emisor ==========`
	`Ingrese una trama en binario: 1101 0101 1101 0010 1000 1100 1011 0011 0101`
	`Trama Codificada: 1101 0101 1110 1001 0100 0110 0100 1100 1110 1001 00`
		1101 0101 1110 1001 0100 0110 0100 1100 1110 1001 00	`========== Hamming Receptor ==========`
			`Trama: 1101 0101 1110 1001 0100 0110 0100 1100 1110 1001 00`
			`No error detectado en la trama.`
			`Trama 1101010111101001010001100100110011101100100`
			Sin errores.

1011 1011 1010 1110 1100 1001 0011 0101 1110	`========== Hamming Emisor ==========`
	`Ingrese una trama en binario: 1011 1011 1010 1110 1100 1001 0011 0101 1110`
	`Trama Codificada: 1011 1011 1011 0111 0110 0100 1000 1101 0101 1100 00`
		1011 1011 1011 0111 0110 0100 1000 1101 0101 1100 00	`========== Hamming Receptor ==========`
			`Trama: 1011 1011 1011 0111 0110 0100 1000 1101 0101 1100 00`
			`No error detectado en la trama.`
			`Trama 1011101110110111011001001000110101011100000`
			Sin errores.

1100 0101 0110 1100 1010 0110 0000 1011 0011	`========== Hamming Emisor ==========`
	`Ingrese una trama en binario: 1100 0101 0110 1100 1010 0110 0000 1011 0011`
	`Trama Codificada: 1100 0101 0111 0110 0101 0011 0010 0010 1110 0101 10`
		1100 0101 0111 0110 0101 0011 0010 0010 1110 0101 10	`========== Hamming Receptor ==========`
			`Trama: 1100 0101 0111 0110 0101 0011 0010 0010 1110 0101 10`
			`No error detectado en la trama.`
			`Trama 1100010101110110010100110010001011100010110`
			Sin errores.

Posteriormente, se quiso determinar la habilidad de los algoritmos para detectar errores. Para esto, se alteró manualmente un bit de tres tramas ya procesadas por los emisores. Esto con la intención de poder determinar si el algoritmo CRC-32 es capaz de detectar dichos errores, y si el algoritmo Código de Hamming es capaz de detectarlos y corregirlos. Como se puede observar a continuación (en la *tabla 3* y *tabla 4*), ambos algoritmos lograron identificar correctamente que las tramas transmitidas sí presentaban errores. Asimismo, el Código Hamming logró corregir correctamente los errores.

***Tabla 3**. Pruebas con CR-32 (con un bit erróneo)*

Trama (Original)	***Resultado Emisor***	***Entrada Receptor***	***Resultado Receptor***

1111 0101 1111 0010 1000 1100 1011 0011 1111	`---------- CRC32 EMISOR ----------`
	`Ingrese una trama en binario: 1111 0101 1111 0010 1000 1100 1011 0011 1111`
	`Trama codificada: 11110101111001010001100101100111111000111100110111011001000001010011`
	`stefanoaragoni@Stefanos-MacBook-Pro lab2_redes %`
		1111 0101 1110 0010 1000 1100 1011 0011 1111 0001 1110 0110 1110 1101 1000 0010 1001	`---------- CRC32 Receptor ----------`
			`Trama: 1111 0101 1110 0010 1000 1100 1011 0011 1111 0001 1110 0110 1110 1101 1000 0010 1001`
			`Resultado: Error en la trama | 001111110000011011011011000010`
			`stefanoaragoni@Stefanos-MacBook-Pro lab2_redes %`
			Con error.

1000 1011 1000 1110 1100 1001 0011 0101 1000	---------- CRC32 EMISOR ---------- Ingrese una trama en binario: 1000 1011 1000 1110 1100 1001 0011 0101 1000 Trama codificada: 10001011100011101100100100111000111011000011101101000 stefanoaragoni@Stefanos-MacBook-Pro lab2_redes %	1000 1011 1000 1110 1100 1001 0010 0101 1000 1011 1111 0001 1101 1000 0011 1011 0100	---------- CRC32 Receptor ---------- Trama: 1000 1011 1000 1110 1100 1001 0010 0101 1000 1011 1111 0001 1101 1000 0011 1011 0100 Resultado: Error en la trama | 110010101100111100101110010101010 stefanoaragoni@Stefanos-MacBook-Pro lab2_redes % *Con error.*
1001 0101 1001 1100 1010 0110 0000 1011 1001	---------- CRC32 EMISOR ---------- Ingrese una trama en binario: 1001 0101 1001 1100 1010 0110 0000 1011 1001 Trama codificada: 10010101100111001010011000001011100011011001010110000111111101110000111100 stefanoaragoni@Stefanos-MacBook-Pro lab2_redes %	1101 0101 1001 1100 1010 0110 0000 1011 1001 1010 1100 0111 1111 0111 0000 1111 1100	---------- CRC32 Receptor ---------- Trama: 1101 0101 1001 1100 1010 0110 0000 1011 1001 1010 1100 0111 1111 0111 0000 1111 1100 Resultado: Error en la trama | 101101000011111101110100001011011 stefanoaragoni@Stefanos-MacBook-Pro lab2_redes % *Con error.*

***Tabla 4**. Pruebas con Código Hamming (con un bit erróneo)*

Trama (Original)	***Resultado Emisor***	***Entrada Receptor***	***Resultado Receptor***
1111 0101 1111 0010 1000 1100 1011 0011 1111	---------- Hamming Emisor ---------- Ingrese una trama en binario: 1111 0101 1111 0010 1000 1100 1011 0011 1111 Trama Codificada: 1111 0101 1101 1001 0100 0110 0110 1100 11 11 1111 01	1110 0101 1101 1001 0100 0110 0110 1100 1111 1111 01	Trama: 1110 0101 1101 1001 0100 0110 0110 1100 1111 1111 01 Errores Detectado en la posicion: 3 (de izquieda a derecha comenzando por 0). Error detectado y corregido en la trama. Trama correcta 1111010111010010010001100110111001111111101 *Con error; corregido correctamente.*
1000 1011 1000 1110 1100 1001 0011 0101 1000	---------- Hamming Emisor ---------- Ingrese una trama en binario: 1000 1011 1000 1110 1100 1001 0011 0101 1000 Trama Codificada: 1000 1011 1010 0111 0110 0100 1010 1101 01 01 0000 11	1000 1011 1010 0111 0110 0100 1010 1101 0101 1000 11	---------- Hamming Receptor ---------- Trama: 1000 1011 1010 0111 0110 0100 1010 1101 0101 1000 11 Errores Detectado en la posicion: 36 (de izquieda a derecha comenzando por 0). Error detectado y corregido en la trama. Trama correcta 1000101110100111011001001010110101010000011 *Con error; corregido correctamente.*
1001 0101 1001 1100 1010 0110 0000 1011 1001	Ingrese una trama en binario: 1001 0101 1001 1100 1010 0110 0000 1011 1001 Trama Codificada: 1001 0101 1010 1110 0101 0011 0010 0010 11 01 0011 00	1001 0101 1010 1110 0101 1011 0010 0010 1101 0011 00	---------- Hamming Receptor ---------- Trama: 1001 0101 1010 1110 0101 1011 0010 0010 1101 0011 00 Errores Detectado en la posicion: 20 (de izquieda a derecha comenzando por 0). Error detectado y corregido en la trama. Trama correcta 1001010110101110010100110010001011010011 00 *Con error; corregido correctamente.*

A partir de este procedimiento, se propuso analizar con más profundidad la robustez de los algoritmos al intentar detectar (y corregir) más de un error. Por dicho motivo, se repitió la metodología de la última prueba, pero se modificó más de un bit. Como se puede observar a continuación (en la *tabla 5* y *tabla 6*), ambos algoritmos lograron identificar correctamente que las tramas transmitidas sí presentaban errores. Sin embargo, el Código Hamming no logró corregir los errores

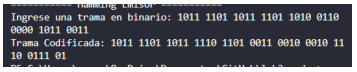
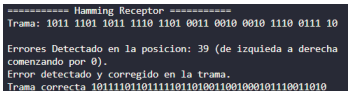
adecuadamente, pues el algoritmo de Hamming solo permite corregir errores de un solo bit. Por tal razón, el comportamiento erróneo era esperado.

***Tabla 5**. Pruebas con CR-32 (con dos o más bits erróneos)*

Trama (Original)	***Resultado Emisor***	***Entrada Receptor***	***Resultado Receptor***
1100 0011 1100 0011 1000 1100 1011 0011 0000	---------- CRC32 EMISOR ---------- Ingrese una trama en binario: 1100 0011 1100 0011 1000 1100 1011 0011 0000 Trama codificada: 11000011110000011100011 00101100110000001010110100101110001111000 10100 stefanoaragoni@Stefanos-MacBook-Pro lab2_redes %	1100 0011 1100 0011 1000 1100 1011 0011 0011 0010 1011 0100 1011 1000 1111 0001 0100	---------- CRC32 Receptor ---------- Trama: 1100 0011 1100 0011 1000 1100 1011 0011 0011 0010 1011 0100 1011 1000 1111 0001 0100 Resultado: Error en la trama | 1011011011 10001000001100100000010 stefanoaragoni@Stefanos-MacBook-Pro lab2_redes % *Con errores.*
1010 1010 0101 0101 1100 1001 0011 0101 0001	---------- CRC32 EMISOR ---------- Ingrese una trama en binario: 1010 1010 0101 0101 1100 1001 0011 0101 0001 Trama codificada: 10101010010101011001 00100110101000101001110011001010111111 01010001 stefanoaragoni@Stefanos-MacBook-Pro lab2_redes %	1010 1010 0100 1101 1100 1001 0011 0101 0001 0100 1110 0110 0101 0111 1111 0101 0001	---------- CRC32 Receptor ---------- Trama: 1010 1010 0100 1101 1100 1001 0011 0101 0001 0100 1110 0110 0101 0111 1111 0101 0001 Resultado: Error en la trama | 0010000010 00001011011011001000011 stefanoaragoni@Stefanos-MacBook-Pro lab2_redes % *Con errores.*
1011 1101 1011 1101 1010 0110 0000 1011 0011	---------- CRC32 EMISOR ---------- Ingrese una trama en binario: 1011 1101 1011 1101 1010 0110 0000 1011 0011 Trama codificada: 10111101101111010110 01100000101100110000011011011011001011 00111100 stefanoaragoni@Stefanos-MacBook-Pro lab2_redes %	1011 0000 1011 1101 1010 0110 0000 1011 0011 0000 0110 1101 1011 0010 1100 0011 1100	---------- CRC32 Receptor ---------- Trama: 1011 0000 1011 1101 1010 0110 0000 1011 0011 0000 0110 1101 1011 0010 1100 0011 1100 Resultado: Error en la trama | 1101110111 00001100011011101111000 stefanoaragoni@Stefanos-MacBook-Pro lab2_redes % *Con errores.*

***Tabla 6**. Pruebas con Código Hamming (con dos o más bits erróneos)*

Trama (Original)	***Resultado Emisor***	***Entrada Receptor***	***Resultado Receptor***
1100 0011 1100 0011 1000 1100 1011 0011 0000	---------- Hamming Emisor ---------- Ingrese una trama en binario: 1100 0011 1100 0011 1000 1100 1011 0011 0000 Trama Codificada: 1100 0011 1100 0001 1100 0110 0100 1100 11 10 0010 11	1101 0011 1100 0001 1100 0110 0110 1100 1110 0010 11	---------- Hamming Receptor ---------- Trama: 1101 0011 1100 0001 1100 0110 0110 1100 1110 0010 11 Errores detectados en la trama. Se descarta la trama por errrores no corregibles. *Error no corregible.*
1010 1010 0101 0101 1100 1001 0011 0101	---------- Hamming Emisor ---------- Ingrese una trama en binario: 1010 1010 0101 0101 1100 1001 0011 0101 0001 Trama Codificada: 1010 1010 0110 1010 1110 0100 1010 1101 01 10 0011 01	0101 1010 0110 1010 1110 0100 1010 1101	---------- Hamming Receptor ---------- Trama: 0101 1010 0110 1010 1110 0100 1010 1101 0110 0011 01 Errores Detectado en la posicion: 30 (de izquieda a derecha comenzando por 0). Error detectado y corregido en la trama. Trama correcta 0101101001101010111001001010111110001101 *Error no corregido correctamente.*

0001		0110 0011 01	
1011 1101 1011 1101 1010 0110 0000 1011 0011		1011 1101 1011 1110 1101 0011 0010 0010 1110 0111 10	 *Error no corregido correctamente.*

Finalmente, se modificó la trama especialmente para que el algoritmo del lado del receptor no fuera capaz de detectar errores. Esto con el propósito de poder detectar las debilidades de los algoritmos. Como se puede observar a continuación (en la *tabla 7* y *tabla 8*), fue necesario modificar más de un bit para que dichos errores pasaran desapercibidos.

***Tabla 7**. Prueba con CR-32 (error no detectable)*

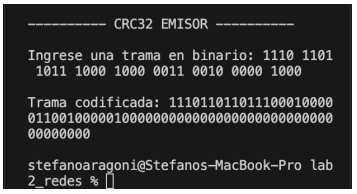
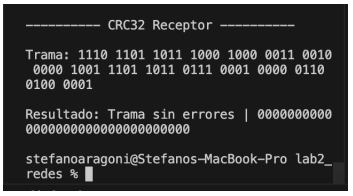
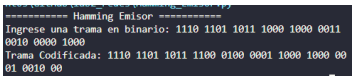
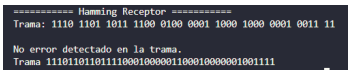
Trama (Original)	***Resultado Emisor***	***Entrada Receptor***	***Resultado Receptor***
1110 1101 1011 1000 1000 0011 0010 0000 1000		1110 1101 1011 1000 1000 0011 0010 0000 1001 1101 1011 0111 0001 0000 0110 0100 0001	 *Sin errores.*

***Tabla 8**. Prueba con Código Hamming (error no detectable)*

Trama (Original)	***Resultado Emisor***	***Entrada Receptor***	***Resultado Receptor***
1110 1101 1011 1000 1000 0011 0010 0000 1000		1110 1101 1011 1100 0100 0001 1000 1000 0001 0011 11	 *Sin errores.*

5. Discusión

El presente informe presenta un estudio detallado sobre la implementación y evaluación de dos algoritmos de detección y corrección de errores: CRC-32 y Código de Hamming. Los objetivos planteados se alcanzaron con éxito, permitiendo comprender el funcionamiento de ambos algoritmos a profundidad, así como sus ventajas y desventajas.

Durante la presente práctica se pudo observar como el Código de Hamming ofrece una ventaja en términos de velocidad y uso eficiente de recursos en comparación con CRC-32. Esto debido a que, a diferencia de CRC-32, no agrega una gran cantidad de bits adicionales a la trama original. Sin embargo, cabe destacar que este algoritmo únicamente puede corregir un único bit de error por trama. Esto quiere decir que si hay más de un bit con error, dicho algoritmo los puede detectar. Sin embargo, no podrá corregir correctamente la trama. Como se puede observar en el apartado de resultados (*figura 6*), al cambiar más de un bit, el algoritmo puede detectar el error pero no sabe como corregirlo, o simplemente lo corregía de manera incorrecta.

Por tal razón, se puede concluir que el algoritmo Código Hamming sería muy útil en aplicaciones donde no hay mucho ruido presente. Esto debido a que, como se mencionó anteriormente, únicamente puede corregir un único bit. A pesar de esta debilidad, el Código Hamming puede ser ventajoso cuando se requiere una corrección precisa y ágil, ya que este algoritmo no tiene tanto *overhead* como CRC-32. Asimismo, como se muestra en la sección de resultados, de todas las pruebas realizadas, este algoritmo siempre fue capaz de detectar la presencia de errores.

Por otro lado, el algoritmo CRC-32 demostró ser igual de robusto en la detección de errores, logrando identificar la presencia de errores en todas las pruebas realizadas; esto incluyendo pruebas con 1, 2 y hasta 4 bits erróneos por trama. Algo que sí se pudo notar, es que este algoritmo añade varios bits adicionales a la hora de transmitir el mensaje. Aunque esto puede variar dependiendo del grado del polinomio utilizado en CRC, la presencia de tantos bits adicionales puede ralentizar el proceso de transmisión de mensaje. Cabe destacar que un grado polinómico bajo aceleraría el proceso de transmisión de mensajes, sin embargo, reduciría la robustez del mismo algoritmo. Por otro lado, también es importante mencionar que este algoritmo no es capaz de corregir errores como el Código de Hamming, lo que lo hace más adecuado para aplicaciones que priorizan la integridad de la información y la detección de errores, sin necesidad de corrección.

En la última prueba, se encontró que resultaba altamente complicado identificar una trama modificada que el algoritmo del receptor no pudiese detectar como errónea. Esto se debía a que, para evitar la detección de errores, era necesario generar una trama válida que mantuviera similitud en sus valores binarios con la original. En ambos casos analizados (*figura 7* y *figura 8*), las tramas "modificadas" demostraron ser válidas al representar valores binarios cercanos a los de la trama original. Cabe destacar que esto en realidad no fue una falla en la

implementación del algoritmo, sino una debilidad en sí del algoritmo. Si una trama sufre cambios y se transforma en otra trama válida, los algoritmos no lo podrán detectar.

En conclusión, en esta práctica se comprendió a profundidad la importancia de la elección del algoritmo de detección y corrección de errores adecuado y como esta depende de restricciones, limitaciones y utilidades específicas de la aplicación. Por ende, la elección entre el Código de Hamming y el algoritmo CRC-32 dependerá de las necesidades específicas de la aplicación y las restricciones del sistema. Ambos algoritmos son valiosos en sus respectivos contextos y ofrecen diferentes enfoques para abordar la problemática de errores en la transmisión de datos

6. Comentario

En nuestro parecer, esta actividad nos ayudó a comprender los retos asociados con la transmisión de mensajes en un entorno propenso a ruido y errores. Al enviar un mensaje, se debe de garantizar que el mensaje recibido es lo que el emisor originalmente quería transmitir. A través de los dos algoritmos implementados, se pudo determinar cómo estos son capaces de alertar a un receptor si el mensaje recibido pudo haber sido corrompido. Cabe destacar que uno de ellos, el Código Hamming, también fue capaz de corregir errores –bajo ciertas circunstancias–.

Durante la elaboración del primer algoritmo (CRC-32), lo más difícil fue garantizar el funcionamiento correcto de la implementación. Esto debido a que se estaba comparando el *checksum* generado por nuestra implementación con los resultados de calculadoras CRC-32 *online*. Ya que las páginas en línea implementan versiones diferentes de CRC-32, no se estaba obteniendo los resultados esperados con los programas desarrollados. Sin embargo, nosotros no sabíamos esto hasta que confirmamos con el profesor Jorge Yass. Él nos indicó que el algoritmo CRC explicado en clase era la versión básica y que no íbamos a obtener el mismo resultado en estas calculadoras en línea.

Durante la implementación del algoritmo de Hamming la parte más compleja fue la detección y corrección de errores. Esto debido a que no nos recordábamos que Hamming sólo podía corregir un bit y perdimos mucho tiempo ideando la forma de corregir más de un bit. Luego de investigar más al respecto, fue posible una implementación más certera y correcta de este algoritmo.

A la hora de realizar las pruebas del algoritmo CRC-32, se pudo notar que siempre se lograba detectar la presencia de errores. Por tal motivo, fue un proceso muy largo poder descifrar qué *bits* cambiar para que el programa no los detectara. Esto fue algo bueno, ya que se tenía la certeza de que se había implementado el algoritmo correctamente y que este tenía un buen rendimiento.

Por otro lado, al realizar pruebas con el Código de Hamming, por la naturaleza del algoritmo, este solo era capaz de detectar la presencia de errores de un solo bit. Cuando se cambiaban dos o más, el algoritmo detectaba la presencia de

errores pero no lograba corregirlos o simplemente lo corregía de manera incorrecta. Esto fue muy interesante de observar, ya que nos permitió determinar las ventajas y desventajas de cada algoritmo.

7. Conclusiones

1. El objetivo principal de de esta práctica fue logrado pues se logró comprender a detalle el funcionamiento de los algoritmos de detección y corrección de errores de Hamming y CRC-32.
2. El Código de Hamming es ventajoso ya que requiere menos recursos y tiene menos *overhead* que CRC-32. Esto se debe a que agrega una cantidad menor de bits adicionales para la corrección de errores.
3. Se puede concluir que esta versión del algoritmo de Hamming se utiliza principalmente en aplicaciones donde el tamaño del mensaje es pequeño y la corrección precisa de un único bit es suficiente. Esto debido a su limitación de únicamente poder corregir un único bit erróneo.
4. El algoritmo CRC-32 agrega varios bits adicionales (mayor *overhead*) para la detección de errores. Cabe destacar que esta adición de bits puede ralentizar el proceso de transmisión de mensajes. Sin embargo, dicha desventaja es compensada por su capacidad para detectar múltiples errores en una trama de forma precisa.
5. Debido al funcionamiento de CRC, este tiene la ventaja que se puede cambiar el grado del polinomio dependiendo de las necesidades específicas de un sistema. Esto permite poder alterar la cantidad de bits adicionales añadidos (*overhead*), lo cual puede ser útil si se desea añadir más robustez o si se desea acelerar el proceso de transferencia de mensajes.

8. Bibliografía

Patel, J. (2023). *Cyclic redundancy check and modulo-2 division*. GeeksforGeeks. https://www.geeksforgeeks.org/modulo-2-binary-division/

Williams, R. (1993). CRC. http://chrisballance.com/wp-content/uploads/2015/10/CRC-Primer.html

Invarato, R. (2017, 12 agosto). Código de Hamming: detección y corrección de errores - Jarroba. Jarroba. https://jarroba.com/codigo-de-hamming-deteccion-y-correccion-de-errores/

Wright, G. (2022b). Hamming code. WhatIs.com. https://www.techtarget.com/whatis/definition/Hamming-code#:~:text=Hamming%20code%20is%20an%20error,data%20is%20stored%20or%20transmitted.